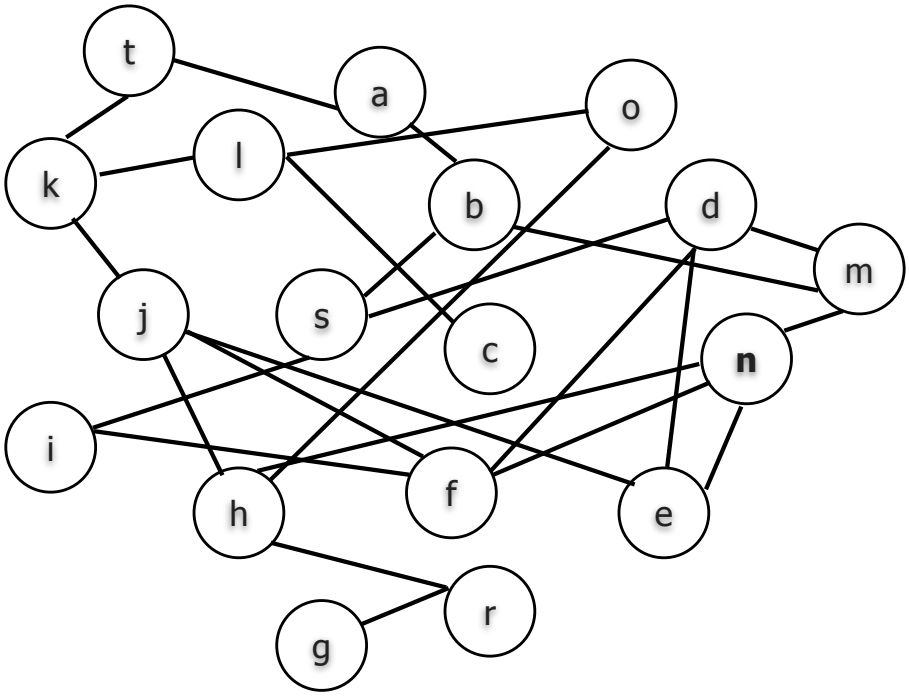
CSE421: Introduction to Algorithms January 16, 2022

Homework 2

Yin Tat Lee Due: Jan 19, 2022 (before class)

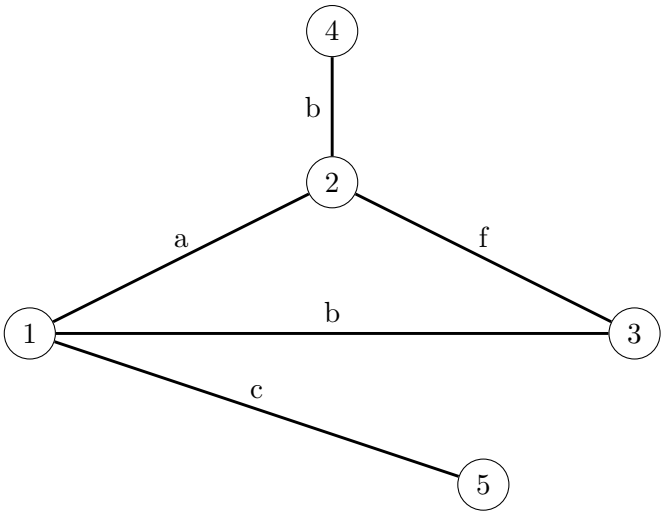
1. (10 Marks) Is the following graph bipartite? If it is, show how to 2-color the graph. If it is not bipartite, show an odd cycle. (It might be easier to work with the printed copy of the graph).



2. (10 Marks) Recall how words are ordered in the dictionary. Suppose we are given an undirected graph $G = (V, E)$ where each edge is labeled with a lowercase letter from "a" to "z" such that each vertex has at most one incident edge labeled with the same letter. For any path in the graph, we can create a string by taking the letters along the edges of the path. Note that each string can describe at most one path starting at a given vertex. Given two vertices u and v, let the *shortest dictionary path* from u to v be the simple path whose corresponding string comes first in dictionary order.

   Design an algorithm that, given a connected graph G with n vertices and m edges and vertex u, outputs the vertices of G in the dictionary order of the strings corresponding to their shortest dictionary paths starting from u. Your algorithm must run in $O(n + m \log n)$ time.

   For example, consider the following graph,



Starting from vertex 1, the shortest dictionary paths to each of the other vertices are:

- 1: `""`
- 2: `"a"`
- 3: `"af"`
- 4: `"ab"`
- 5: `"c"`

Our algorithm would output the ordering `1, 2, 4, 3, 5`.

Remark: Using the fact that there are only 26 lower case alphabets, you can get an runtime of $O(n)$ (smaller than what we asked).

3. (10 Marks) Given a graph $G = (V, E)$ with n vertices and m edges. Given a vertex s. Let $k(u)$ be the number of distinct shortest paths from s to u. Assume all edges are unit length. Give an algorithm that computes $k(u)$ for each vertex u of G in time $O(n + m)$. Justify the running time bound of your algorithm.

   (Note that there can be exponentially many shortest paths between two vertices. Therefore, it is impossible to list out all these paths in linear time.)

4. **(Extra Credit)** Given a tree with n vertices. We want to design a data structure with the following operations:

   (a) $\mathsf{Init}()$: set the value v_e to 0 for each edge e in time $O(n)$.

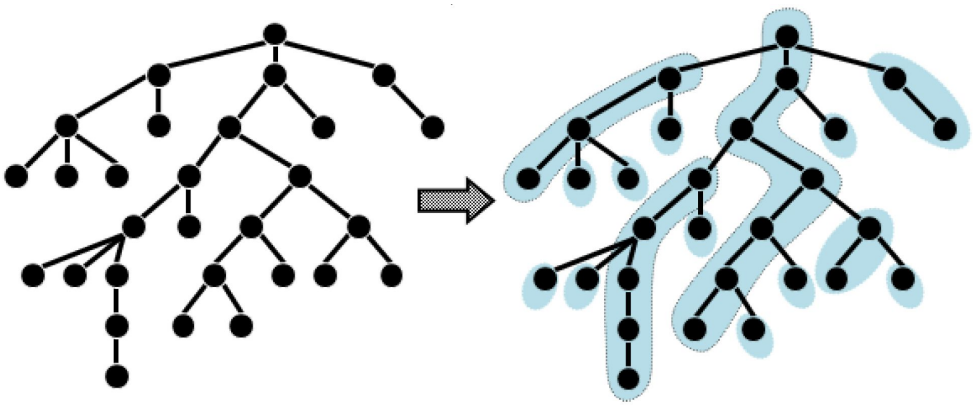
   (b) $\mathsf{Set}(e, u)$: set the value of v_e for edge e to u in time $O(\log^{O(1)} n)$.

   (c) $\mathsf{Query}(s, t)$: output $\sum_{e \in P} v_e$ where P is the path from s to t on the tree in time $O(\log^{O(1)} n)$.

   Hints: (There are multiple ways to solve this problem, Here is one of them.)

   (a) Usually problems are simpler for paths. You can explain how to solve the problem on paths first.

(b) In the following example, for each vertex, I colored the edge leading to the largest subtree.



Show that each path on the tree consists only $O(\log n)$ segments of the colored path.

(c) The solution of this homework is a bit long. Feel free to just explain the key ideas.

Finally, we note that similar data structures can be designed for any associative operation, such as max.